#13
11/25/02

Patent
Docket No. 10982056-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

APPEAL NO. _____

In re Application of:
Carter et al

Serial No. 09/272,810
Filed: March 19, 1999

For: **NETWORK SEVER USING LOCAL INFORMATION TO DETECT TIMED-OUT CLIENT REQUESTS**

APPELLANT'S BRIEF ON APPEAL

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents Washington, D.C. 20231 on November 9, 2002.

HUGH P. GORTLER, Reg. No. 33,890

Hugh P. Gortler, Esq.

Hewlett-Packard Company
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

(949) 454-0898

INDEX

1. REAL PARTY IN INTEREST

The real party in interest is the assignee, Hewlett-Packard Company.

2. RELATED APPEALS AND INTERFERENCES

Appellant is not aware of any appeals or interferences that would have a bearing on the Board's decision in the pending appeal.

3. STATUS OF CLAIMS

Claims 1-21 are rejected under 35 U.S.C. §102(e) as being anticipated by Huras et al. for the reasons given in the Office Action dated May 15, 2002 and made final. Claims 1-21, listed in Appendix A, are being appealed.

4. STATUS OF AMENDMENTS

No amendment was filed subsequent to final rejection.

5. SUMMARY OF THE INVENTION

A web site may be accessed by a "client" such as a personal computer running a web browser program that is capable of connecting to the Internet. A user enters a Uniform Resource Locator ("URL") of the web site into the client. As a result, the client attempts to make a connection with the server and, if successful, sends a client request (e.g. for web content) to the server. Under ideal conditions, the client receives a response back from the server and displays the requested web content.

A popular web site might receive large bursts of client requests at any given time. A high-performance, high-capacity HTTP (a.k.a. "web") server is typically configured to process a limited number of these requests concurrently. Any additional pending requests (and their associated connection information) are

temporarily buffered by the server in its "listen queue." In servicing client requests, web applications running on the web server might provide both static and dynamic content, performing complex database queries and other data-manipulation. These can lead to long delays in responding to client requests. Congested and overloaded Internet routes only add to the delays.

Long delays typically cause users to cancel and possibly resubmit their requests. Since timed-out requests are not removed from the listen queues of current web servers, their processing could lead to a substantial expenditure of server resources. An overloaded web server could end up processing a lot of "dead" timed-out requests. While the web server is processing these timed-out requests, it is expending its resources on useless work instead of devoting its resources toward "still-vital" requests.

The present invention addresses the problem of timed-out client requests. Reference is made to Figure 5 and claim 1 of the application (Figure 5 is attached hereto as Appendix B). A server handles a network connection by examining local server information to determine whether a client-to-server channel is still established (310); and aborting response preparation to a client request if the client-to-server channel is determined to be no longer established (312, 316).

Such a network server can detect timed-out client requests with little computational overhead. Therefore, the server can afford to check whether a response to a client's request is still needed before expending server resources on generating a response to that request. The amount of computational resources spent on processing dead requests is reduced and, consequently, server efficiency is increased. Moreover, protection against request-timeout livelock is provided.

6. THE ISSUE

Whether Huras et al. teach or suggest a server that aborts response preparation to a client request if a client-to-server channel is determined to be no longer established.

7. GROUPING OF CLAIMS

Claims 1-21 stand or fall together.

8. ARGUMENT

CLAIMS 1-21 SHOULD BE ALLOWED OVER HURAS ET AL. ALONE BECAUSE HURAS ET AL. DO NOT TEACH OR SUGGEST A SERVER THAT ABORTS RESPONSE PREPARATION TO A CLIENT REQUEST IF A CLIENT-TO-SERVER CHANNEL IS DETERMINED TO BE NO LONGER ESTABLISHED

Figs. 2A and 2B of Huras et al. are attached as Appendix C. These figures illustrate steps taken by a client process and a server process. The flow of steps taken by the server process depends upon a "valid request flag."

As shown in Fig. 2A and described in col. 7 lines 37-44, the client process receives a request from an application (block 10), writes the request data to shared memory (block 15), sets the valid request flag to "true" (block 20), and then sets a send semaphore (block 25). Once the send semaphore is set, the operating system ends a wait function executed by the server process (block 9, Fig. 2B) and immediately clears the just-posted send semaphore (col. 7, lines 57-61). The operating system puts the server process on a queue of processes that are ready to run.

If the client process terminates while the server process is on the queue, the server process still prepares a response. Response preparation is not

aborted. Even though the operating system posts the send semaphore (col. 8, lines 59-62), **it does not alter the value of the request flag, which remains as "true."** The server process, when next allowed to run, sees the "true" request flag as indicating a valid request. Because the server process sees a true request flag, it prepares a response.

Support is provided by Fig. 2B and col. 7, lines 57+ of Huras et al, which describes the server process. Once the wait state has ended (block 9), the server process proceeds by setting the valid request flag to "false" (block 55), reading the request data in shared memory (block 60), ***processing the request*** (block 65), and writing the response data to shared memory (block 70). After this response preparation, the server process proceeds through blocks 75, 9, and 50. The server process now sees a false valid request flag and proceeds to block 80, where it frees up resources and terminates. Thus, the server process prepares a response and then deallocates resources, even if the client process terminates before response preparation began.

On col. 8, lines 59-67, Huras et al. state that processing the request (block 65) can be bypassed if the client process is terminated, either because the application has been completed its processing or the operating system terminated the client for some reason ("In this case, the valid request flag will not have been set to true by the client process"). However, this passage does not teach or suggest that response preparation is aborted due to a determination that the client-to-server channel is no longer established.

Pages 2-4 of the final office action are attached as Appendix D. Page 2 of the final office action contends that the col. 7, line 57 to col. 8, line 67 of Huras et al. disclose aborting response preparation to a client request if the client-to-server

channel is determined to be no longer established. It does not. The passages at col. 7, line 57 to col. 8, line 67 describe the flowchart of Fig. 2B. As discussed above, this passage clearly shows that response preparation is not aborted **if the client process terminates**. A response to any pending valid client request is prepared, and then client resources are deallocated, even if the client process terminates before server response preparation started. Request processing can be skipped due to the operating system, but not due to a determination that the client-to-server channel is no longer established

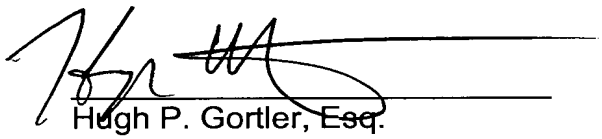
Page 2 of the final office action contends that the Abstract of Huras et al. disclose aborting response preparation to a client request if the client-to-server channel is determined to be no longer established. It does not. The last sentence of the Abstract merely states that the service provider terminates client resources if a flag is not set. As discussed above, Fig. 2B, a response can be prepared before the client resources are deallocated. Moreover, the Abstract says nothing about aborting response preparation due to a determination that the client-to-server channel is no longer established.

According to page 4 of the office action, Huras et al. teach "server process terminate resources allocated to client process to free up system resources if determined that client process has terminated because of any reason." The analysis above of Figs. 2A and 2B clearly indicates this statement does not address the matter of aborting response preparation. More specifically, the statement does not address aborting response preparation to a client request if the client-to-server channel is determined to be no longer established.

9. CONCLUSION

Huras et al. do not disclose that response preparation is aborted if the client-to-server channel is determined to be no longer established. Because they do not disclose all of the limitations recited in claim 1, the '102(e) rejection of claim 1 and its dependent claims 2-7 should be withdrawn. The '102(e) rejection of claims 8-21 should be withdrawn for the same reasons. Moreover, Huras et al. do not suggest aborting response preparation if the client-to-server channel is determined to be no longer established. Therefore, claims 1-21 should be allowed over Huras et al. Appellant respectfully requests the Honorable Board of Patent Appeals and Interferences to hold that claims 1-21 are allowable over Huras et al.

Respectfully submitted,



Hugh P. Gortler, Esq.
Attorney for Appellants
Registration No. 33, 890

Hewlett-Packard Company
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

(949) 454-0898

Date: November 9, 2002

10. APPENDIX

Appendix A. The Claims on Appeal

The appealed claims 1-21 are as follows:

1. In a server, a method of handling a network connection, the network connection including a client-to-server channel and a server-to-client channel, the method comprising:

examining local server information to determine whether the client-to-server channel is still established; and

aborting response preparation to a client request if the client-to-server channel is determined to be no longer established.

2. The method of claim 1, wherein a state of the server-to-client channel is inferred according to whether the client-to-server channel is still established; and wherein the response preparation is aborted if the server-to-client channel is inferred to be closed.

3. The method of claim 1, wherein the server includes a read buffer; wherein the client request is read from the read buffer; and wherein the read buffer is then probed to determine whether the client-to-server channel is still established.

4. The method of claim 1, wherein the server maintains local information about the state of the client-to-server channel; wherein a specific state of the client-to-server channel is determined by examining the local information; and wherein the response preparation is aborted if the local information indicates that the client-to-server channel is in the specific state.

5. The method of claim 4, wherein the client-to-server channel is determined to be no longer established if the local information indicates that the client-to-server channel is in a "CLOSE_WAIT" state.

6. The method of claim 1, wherein the state of the client-to-server channel is determined by polling the local information while a response to the client request is being prepared, whereby response preparation can be aborted while a request is being prepared.

7. The method of claim 1, further comprising generating an interrupt when the client-to-server channel is determined to be no longer established, wherein a response to the client request is processed until the interrupt is generated.

8. A network server comprising:

a processing unit;

a network interface card; and

computer memory programmed to cause the processing unit to examine local server information to determine whether a client-to-server channel is still established; and abort response preparation if the client-to-server channel is determined to be no longer established.

9. The server of claim 8, wherein a state of a server-to-client channel is inferred according to whether the client-to-server channel is still established; and wherein the response preparation is aborted if the server-to-client channel is inferred to be closed.

10. The server of claim 9, further comprising a read buffer; wherein a client request is read from the read buffer; and wherein the read buffer is probed to determine whether the client-to-server channel is still established

11. The server of claim 8, wherein the memory includes local information about a state of the client-to-server channel; wherein a state of the client-to-server channel is determined by examining the local information; and wherein the response preparation is aborted if the local information indicates that the client-to-server channel is in the specific state.

12. The server of claim 11, wherein the client-to-server channel is determined to be no longer established if the local information indicates that the client-to-server channel is in a "CLOSE_WAIT" state.

13. The server of claim 8, wherein a state of the client-to-server channel is determined by polling the local information while a response to the client request is being prepared.

14. The server of claim 8, wherein the memory is programmed with a routine for commanding the processing unit to generate an interrupt when the client-to-server channel is determined to be no longer established, and wherein a response to a client request is processed until the interrupt is generated.

15. A network server comprising:

a processing unit;

first means for maintaining a queue of connections based on connection requests, each network connection including a client-to-server channel and a server-to-client channel;

second means for accepting connections from the queue;

third means for examining local server information to determine whether the client-to-server channel of a given connection from the queue is still established; and

fourth means for aborting response preparation if it is determined that the client-to-server channel of the given connection is no longer established.

16. An article for a network server including a processing unit and a network interface card, the article comprising:

computer memory; and

a server program encoded in the computer memory, the server program commanding the processing unit to accept network connections, each connection having a client-to-server channel and a server-to-client channel; examine local server information to determine whether the client-to-server channel of a given connection from the queue is still established; and abort response preparation if the client-to-server channel of the given connection is determined to be no longer established.

17. The article of claim 16, wherein a state of the server-to-client channel of the given connection is inferred according to whether the corresponding client-to-server channel is still established.

18. The article of claim 16, wherein the memory is further encoded with local information about a state of the given connection; wherein the state of the given connection is determined by examining the local information; and wherein response preparation is aborted if the local information indicates that the client-to-server channel of the given connection is in the specific state.

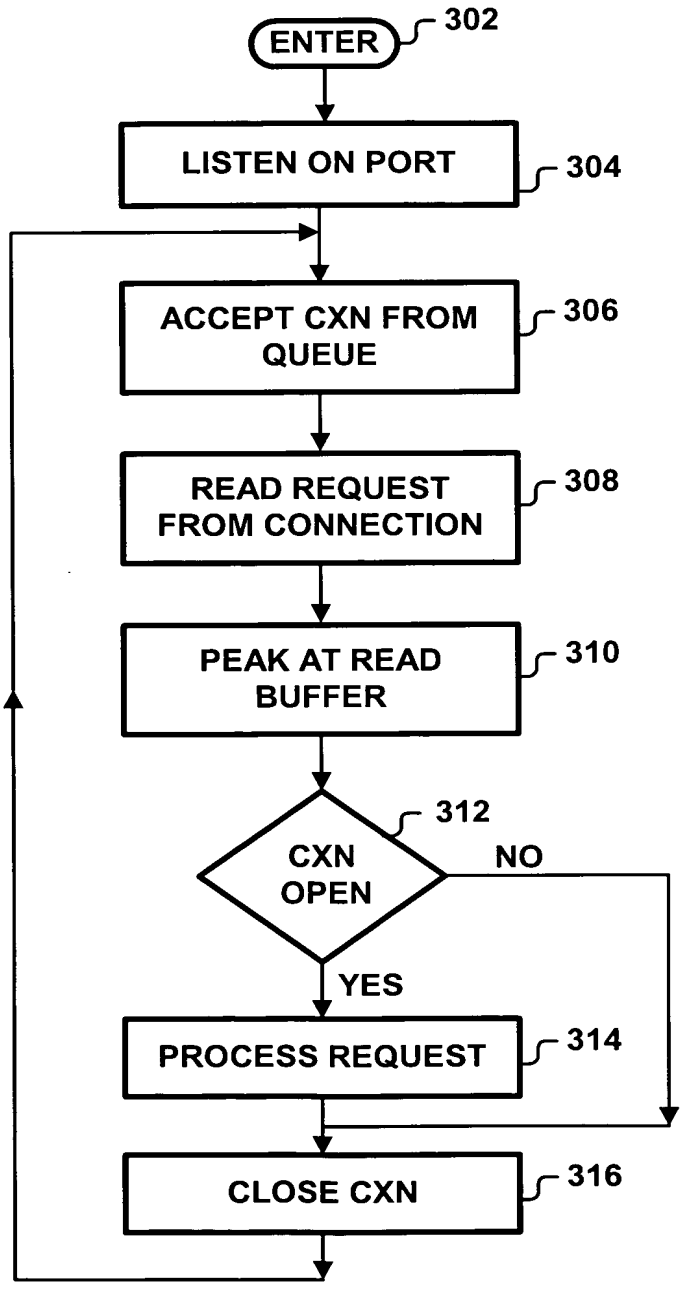
19. The article of claim 16, wherein a state of the client-to-server channel of the given connection is determined by polling the local information, the local information being polled while a response to a client request is being prepared.

20. The article of claim 16, wherein the memory is further encoded with a routine for commanding the processing unit to generate an interrupt when the client-to-server channel of the given connection is determined to be no longer established, and wherein a response to a client request is processed until the interrupt is generated.

21. A computer program for a processing unit, the program comprising instructions for commanding a processing unit to maintain a queue of network connections based on connection requests, the program further comprising instructions for commanding the processing unit to accept connections from the queue; examine local server information to determine whether a client-to-server channel of a given connection from the queue is still established; process a client request associated with the given connection if the client-to-to-server channel of the given connection is still established; and abort response preparation for the associated client request if the client-to-server channel of the given connection is no longer established.

Appendix B

FIG. 5



Appendix C

CLIENT

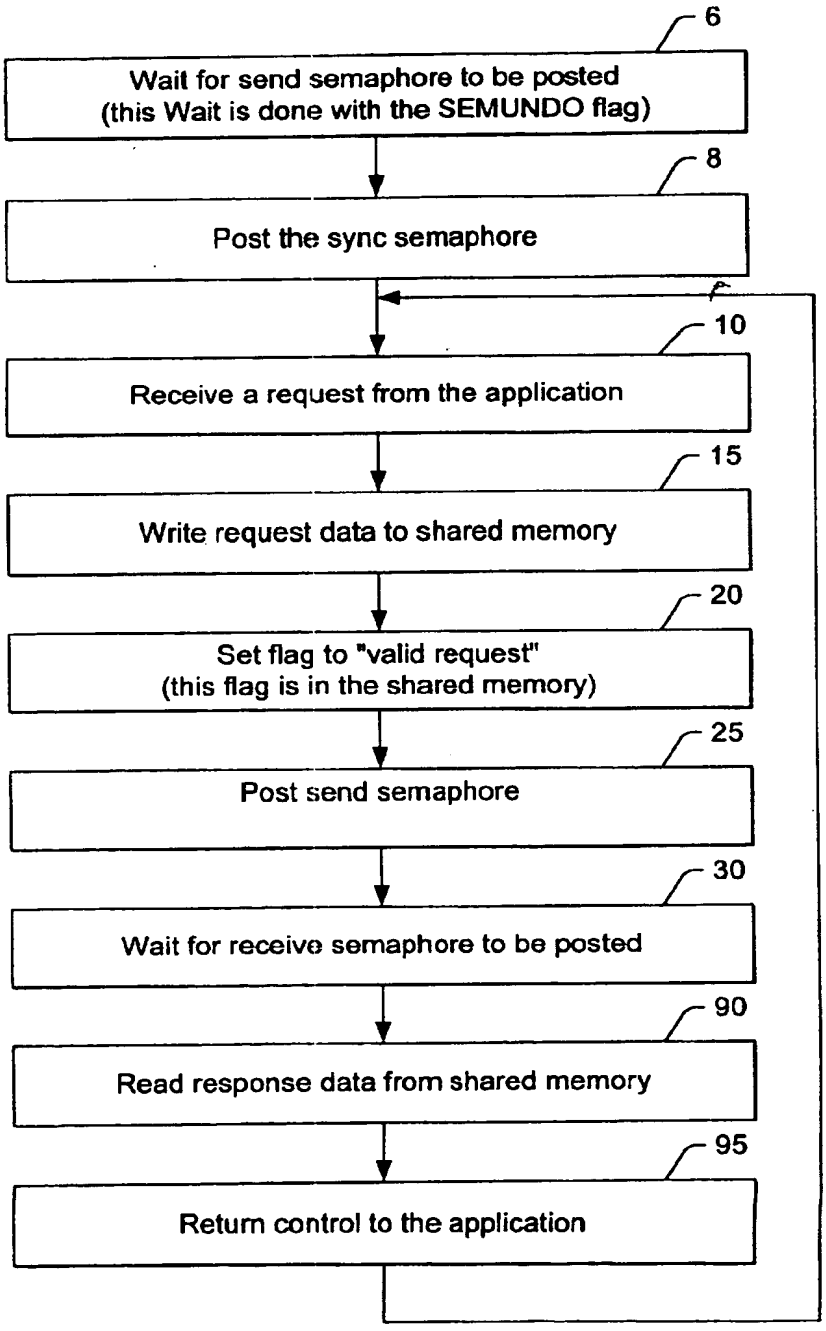


FIG. 2A

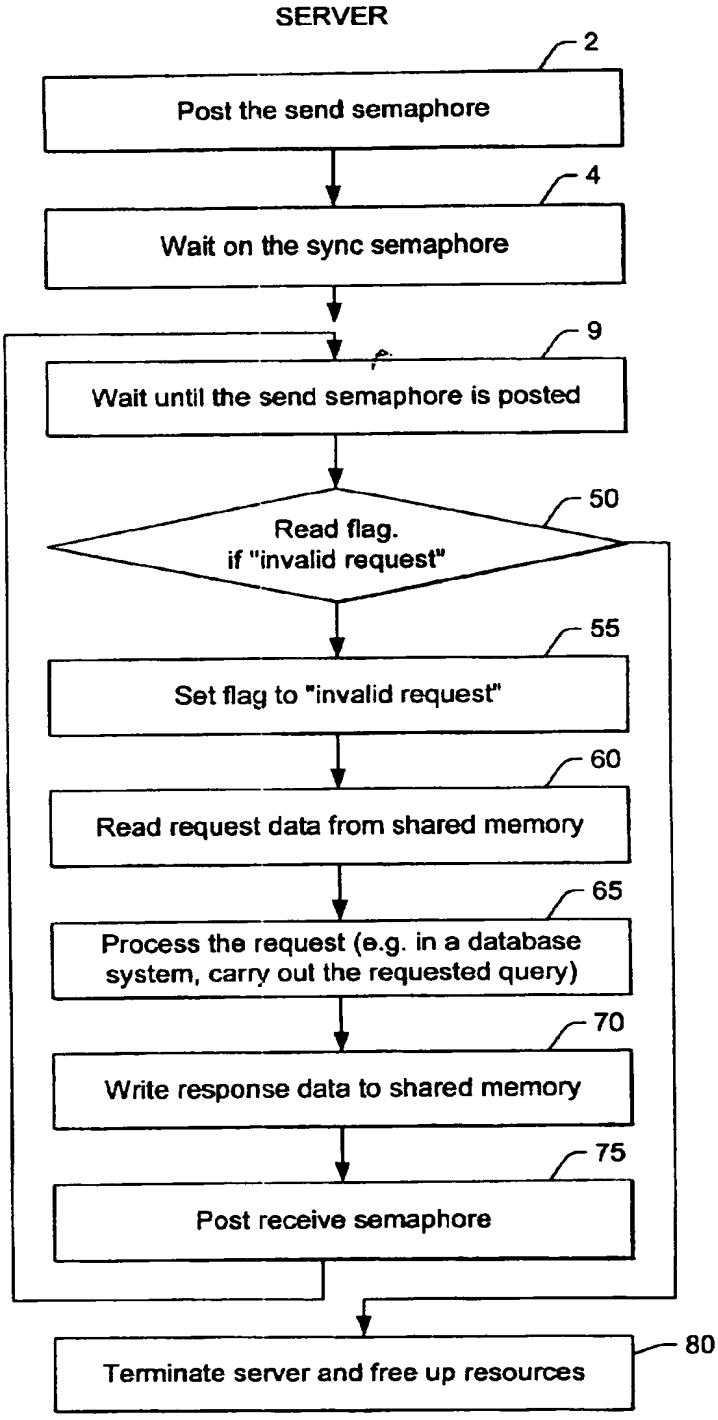

SERVER
2
Post the send semaphore
4
Wait on the sync semaphore
9
Wait until the send semaphore is posted
50
Read flag.
if "invalid request"
55
Set flag to "invalid request"
60
Read request data from shared memory
65
Process the request (e.g. in a database system, carry out the requested query)
70
Write response data to shared memory
75
Post receive semaphore
80
Terminate server and free up resources

FIG. 2B

APPENDIX D

1. The proposed drawing correction filed on 03/05/2002 is approved.

2. Claims 1-21 are presented for examination.

3. The text of those sections of Title 35, U.S. Code § 102 (e) not included in this action can be found in a prior Office Action.

4. Claims 1-21 are rejected under U.S. Code § 102 (e) as being anticipated by **Huras et al. (Huras)** patent no. **6,125,401**.

5. **Huras** was cited as prior art in the last office action.

6. As to claim 1, Huras teaches the invention as claimed, including a method of handling a network connection, the network connection including a client-to-server channel and a server-to-client channel (col. 4 line 39 – col. 8 line 66), the method comprising:

examining local server information to determine whether the client-to-server channel is still established (Abstract; col. 7 line 57 – col. 8 line 21); and

aborting response preparation to a client request if the client-to-server channel is determined to be no longer established (Abstract; col. 7 line 57 – col. 8 line 67).

7. As to claims 2-7, Huras teach the state of the server-to-client channel is inferred after reading from client-to-server channel; a read buffer is being used to determine

whether the client-to-server channel is still established; specific state of the connection is determined by examining local information in the server, "CLOSE_WAIT" state, interrupt. In addition, Huras teaches that polling is being used despite of some disadvantages (col. 1 lines 43-55, col. 4 line 39 - col. 8 line 66).

8. Claims 8-21 have similar limitations as claims 1-7; therefore, they are rejected under the same rationale.

9. In the remarks, applicant argued in substance that

(A) Prior art does not teach a network environment .

As to point (A), Huras teaches a client-server system where a terminal and personal computer connect to a main computer via a network (figure 1; col. 4 lines 39-65).

(B) Prior art does not teach client-to-server channel or a server-to-client channel.

As to point (B), Huras teaches the terminal runs application to interact with a service provider on the main computer thru a channel and vice versa (col. 5 lines 10-23).

(C) Prior art does not teach abort response preparation to a client request if the client-to-server channel is determined to be no longer established.

As to point (C), Huras teaches server process terminate resources allocated to client process to free up system resources if determined that client process has terminated because of any reason (Abstract; col. 7 line 57 – col. 8 line 67).

10. Limitations that are argued by applicant but are not in claimed language are not being considered by Examiner.

11. Applicant's arguments filed on 03/05/2002 have been fully considered but they are not deemed to be persuasive.

12. THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 C.F.R. § 1.136(a).

A SHORTENED STATUTORY PERIOD FOR RESPONSE TO THIS FINAL ACTION IS SET TO EXPIRE THREE MONTHS FROM THE DATE OF THIS ACTION. IN THE EVENT A FIRST RESPONSE IS FILED WITHIN TWO MONTHS OF THE MAILING DATE OF THIS FINAL ACTION AND THE ADVISORY ACTION IS NOT MAILED UNTIL AFTER THE END OF THE THREE-MONTH SHORTENED STATUTORY PERIOD, THEN THE SHORTENED STATUTORY PERIOD WILL EXPIRE ON THE DATE THE ADVISORY ACTION IS MAILED, AND ANY EXTENSION FEE PURSUANT TO 37 C.F.R. § 1.136(a) WILL BE CALCULATED FROM THE MAILING DATE OF THE ADVISORY ACTION. IN NO EVENT WILL THE STATUTORY PERIOD FOR RESPONSE EXPIRE LATER THAN SIX MONTHS FROM THE DATE OF THIS FINAL ACTION.

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Le H. Luu, whose telephone number is (703) 305-9650. The examiner can normally be reached Monday through Friday from 7:00 AM to 4:30 PM.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

AF/2152/#

PATENT APPLICATION

ATTORNEY DOCKET NO. 10982056-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Carter et al.

Confirmation No.: 6119

Application No.: 09/272,810

Examiner: Luu

Filing Date: 3/19/1999

Group Art Unit: 2152

Title: NETWORK SERVER USING LOCAL INFORMATION TO DETECT TIMED-OUT CLIENT REQUESTS

COMMISSIONER FOR PATENTS
Washington, D.C. 20231

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith in **triplicate** is the Appeal Brief in this application with respect to the Notice of Appeal filed on 9/9/2002.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $320.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

( )	one month	$110.00
( )	two months	$400.00
( )	three months	$920.00
( )	four months	$1440.00

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of term is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of $310.00. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

Respectfully submitted,

Carter et al.

By _____

Hugh P. Gortler

Attorney/Agent for Applicant(s)
Reg. No. 33,890

Date: 11/9/2002

Telephone No.: (949) 454-0898

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Washington, D.C. 20231.

Date of Deposit: 11/9/2002

Typed Name: Hugh P. Gortler

Signature: _____